BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平8－263325

(43)公開日　平成８年(1996)10月11日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/30	３１０	7313－5B	Ｇ０６Ｆ　11/30	３１０Ａ

審査請求　未請求　請求項の数15　ＯＬ　（全 12 頁）

(21)出願番号　特願平7－60488

(22)出願日　平成７年(1995)３月20日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　大橋　勝之
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(74)代理人　弁理士　大菅　義之　（外１名）

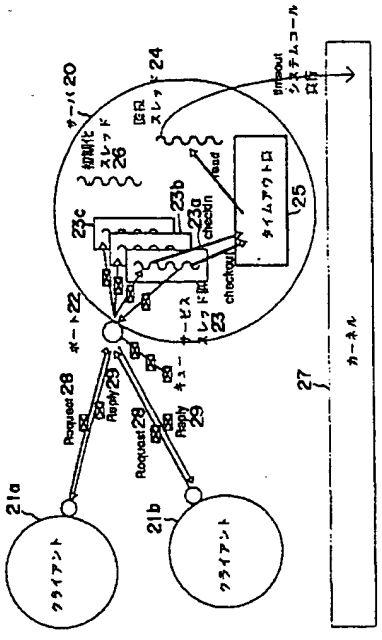
(54)【発明の名称】　サーバ処理装置、サーバ内障害検出装置及びサーバ内障害検出方法

(57)【要約】

【目的】　本発明は、クライアント／サーバモデルのシステムに関し、レスポンスを低下させることなく、複数のクライアントからのリクエストを処理することが可能なサーバ内の障害検出方式を提供することを目的とする。

【構成】　サービススレッド２３ａは、クライアント２１ａからのリクエスト２８を受信すると、要求された処理のタイムアウト時間と自分のスレッドＩＤをタイムアウト表２５に書込む。そして、リクエスト２８の処理を終了すると、タイムアウト表２５内のタイムアウト時間及びスレッドＩＤを消去する。監視スレッド２４は、定期的にタイムアウト表２５からタイムアウト時間を読み出して、現在の時刻と比較することにより、設定されたタイムアウト時間までに処理を終了していないサービススレッドを検出する。タイムアウトしたサービススレッドを発見した場合には、該サービススレッドを強制的に終了させて、クライアント２１にエラーを通知する。

本発明の一実施例の概要を説明する図



1

【特許請求の範囲】

【請求項1】　クライアントから依頼される処理を、複数のスレッドで処理し、その処理結果を該クライアントに返すサーバ処理装置において、

各スレッドによって実行される処理のタイムアウト時間の管理情報が設定されるタイムアウト時間情報管理手段と、

該各スレッドが処理を開始する際に、該タイムアウト時間情報管理手段にそのスレッドの当該タイムアウト時間情報管理情報を設定するタイムアウト時間情報設定手段と、

上記タイムアウト時間情報管理手段に設定されたタイムアウト時間管理情報を用いて、タイムアウトの発生したスレッドを検出するタイムアウト検出手段と、

を備えることを特徴とするサーバ処理装置。

【請求項2】　前記タイムアウト検出手段は、タイムアウトの発生したスレッドを削除すること、

を特徴とする請求項1記載のサーバ処理装置。

【請求項3】　前記タイムアウト検出手段は、スレッドを削除した後に、新たなスレッドを生成すること、

を特徴とする請求項2記載のサーバ処理装置。

【請求項4】　前記タイムアウト検出手段は、タイムアウトの発生したスレッドを強制的に終了させること、

を特徴とする請求項1記載のサーバ処理装置。

【請求項5】　クライアントからのリクエストを受信し、該リクエストの処理を実行するサービス手段と、

該サービス手段が行う処理のタイムアウト時間を管理するタイムアウト時間管理手段と、

前記サービス手段を監視する監視手段とから構成され、

前記サービス手段は、クライアントからリクエストされた処理を開始する前に、該処理のタイムアウト時間を前記タイムアウト時間管理手段に設定し、該処理の終了時にタイムアウト時間管理手段の自分が設定した前記タイムアウト時間を削除し、

前記監視手段は、タイムアウト時間管理手段を参照し、タイムアウト時間内に前記処理が終了しているか否かを検出することを特徴とするサーバ内障害検出装置。

【請求項6】　前記監視手段は、前記検出処理を開始する前にタイムアウト処理を設定し、

該検出処理終了後、該タイムアウト処理を解除することを特徴とする請求項5記載のサーバ内障害検出装置。

【請求項7】　前記サービス手段が設定する前記タイムアウト時間は、設定時の時刻に所定の時間を加えたものであり、

前記監視手段は、タイムアウト時間管理手段から読み出した該タイムアウト時間と、現在の時刻を比較することにより検出処理を行うことを特徴とする請求項5又は6記載のサーバ内障害検出装置。

【請求項8】　前記サービス手段が、複数のサービススレッドから構成され、

2

前記監視手段が、1つの監視スレッドから構成されることを特徴とする請求項5、6又は7記載のサーバ内障害検出装置。

【請求項9】　前記タイムアウト時間管理手段が、スレッドＩＤを格納する「スレッドＩＤ」フィールドおよびタイムアウト時間を設定する「タイムアウト時間」フィールドを有する表から構成されることを特徴とする請求項5、6、7又は8記載のサーバ内障害検出装置。

【請求項10】　前記サービス手段および前記監視手段がプロセスによって構成されることを特徴とする請求項5記載のサーバ内障害検出装置。

【請求項11】　クライアントからのリクエストを受信するポートと、

該リクエストを処理するサービススレッド群と、

スレッド番号とタイムアウト時間を対応づけて格納するタイムアウト表と、

サービススレッド群を監視する監視スレッドから構成されるサーバを有し、

サービススレッド群の各サービススレッドは、前記リクエストされた処理を開始する前に、該処理のタイムアウト時間と自分のスレッド番号をタイムアウト表に記入し、該処理の終了後にタイムアウト表から該タイムアウト時間と自分のスレッド番号を削除し、

監視スレッドが定期的にタイムアウト表を読み出して、タイムアウト時間内に処理を終了していないサービススレッドを検出することを特徴とするサーバ内障害検出装置。

【請求項12】　クライアントからのリクエストを受信し、

該受信したリクエストに対する処理を実行するサービススレッドのスレッド番号とタイムアウト時間をタイムアウト表に設定し、

前記リクエストに対する処理を実行し、

該処理終了後、前記タイムアウト表に設定されたスレッド番号とタイムアウト時間を削除し、

前記タイムアウト表を参照することによりサービススレッドの監視処理を実行することを特徴とするサーバ内障害検出方法。

【請求項13】　前記監視処理を開始する前に、監視処理のタイムアウト処理を設定し、

前記監視処理の終了後、該タイムアウト処理を解除することを特徴とする請求項12記載のサーバ内障害検出方法。

【請求項14】　前記監視処理を終了後、次回の監視処理を始める前に、該監視処理を所定の時間スリープさせることを特徴とする請求項12又は13記載のサーバ内障害検出方法。

【請求項15】　サーバが起動時に、初期化スレッドを生成し、

該初期化スレッドが、タイムアウト表を初期化し、前記

監視処理を行う監視スレッドを生成し、クライアントからリクエストを受け付けるリクエスト受付ポートを作成し、該リクエストに対する処理を行うサービススレッド群を生成することを特徴とする請求項１２、１３又は１４記載のサーバ内障害検出方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は計算機上のクライアント／サーバモデルにおけるサーバ内での障害を検出する装置および方法に係わり、特にクライアントから要求されたサービスの処理中にサーバ内で発生した無限ウエイトや無限ループ等の障害を検出する装置および方法に関する。

【０００２】

【従来の技術】クライアント／サーバモデルは計算機環境の分散化／並列化のために有効な技術であり、関連する技術の開発が望まれている。クライアント／サーバモデルでは、サーバはクライアントからの要求（リクエスト）を受けてサービス処理を行い、その結果をクライアントに返すよう構成される。

【０００３】サーバは、クライアントからのリクエスト処理中に無限ウエイトあるいは無限ループに陥る可能性がある。この場合、サーバだけではなく、サーバからの返答（リプライ）を待っているクライアントも無限ウエイトに陥ってしまい、システム全体が停止してしまう。このような事態が発生するのを避けるため、サーバは、リクエストに対するサービス処理開始前に、システムカーネルがシステムコールとして提供するタイマルーチンを使って該サービス処理のタイムアウト時間を設定していた。サーバが設定したタイムアウト時間内に前記サービス処理が終了しない場合には、システムカーネルがサーバの処理を終了させ、クライアントにサーバのエラーを通知していた。このようにして、サーバの障害がクライアントに伝播してシステム全体が停止する事態が引き起こされるのを防いでいた。

【０００４】図９は、上述した従来技術を説明する図である。同図に示すように、サーバ（サーバプロセス）１０は、クライアント１１ａまたはクライアント１１ｂからのリクエスト１８を受けてサービス処理を行い、返答（リプライ）１９をクライアント１１ａまたはクライアント１１ｂに返す。リクエスト１８およびリプライ１９は、メッセージによって行われる。同図において、中に×が書かれた長方形がメッセージを示している。

【０００５】図９において、クライアント１１ａまたはクライアント１１ｂからのリクエスト１８は、サーバ１０のポート１２で受信され、そのポート１２のキュー９に入れられる。サービススレッド群１３の各サービススレッド１３ａ～１３ｃは、適宜、キュー９からリクエスト１８を取り出して、サービス処理を行う。サービス処理が終了すると、各サービススレッド１３ａ～１３ｃは、リプライ１９をクライアント１１ａまたはクライアント１１ｂに返す。

【０００６】各サービススレッド１３ａ～１３ｃは、前記サービス処理を開始する前にタイムアウト処理のためのシステムコールを実行する。該システムコールを受けて、カーネル１７はタイムアウト処理を行う。すなわち、指定された時間内にサービス処理が終了していない場合は、前記システムコール時に指定される処理ルーチンによって、例えば、タイムアウトしたサービススレッドを消滅させ、リクエスト処理を実行していたクライアントへエラーを通知する等の処理を行う。

【０００７】

【発明が解決しようとする課題】しかしながら、従来の方式では、サーバがクライアントからのリクエストの処理を開始する毎にシステムコールであるタイマルーチンをコールするため、複数のクライアントからの要求を複数のサービススレッドによって並列に処理する場合には、リクエストの増加に伴いシステムコールの実行回数が増加する。システムコールは、サーバ内部の関数呼出しにくらべてオーバヘッドが大きく、システムコールの実行回数が増加すると、サーバのレスポンスが低下してしまう可能性がある。

【０００８】本発明は、サーバ内のサービススレッド数が増加しても、サーバのレスポンス性能の低下を防ぐことができるサーバ内障害検出装置及び方法を提供することを目的とする。

【０００９】

【課題を解決するための手段】図１は、本発明の原理を説明する図（その１）である。この発明に係るサーバ処理装置（以下、第１の発明）は、クライアントから依頼される処理を、複数のスレッドで処理し、その処理結果を該クライアントに返すサーバ処理装置を前提とし、以下の各手段を備える。

【００１０】タイムアウト時間情報管理手段３１は、各スレッドによって実行される処理のタイムアウト時間の管理情報が設定される。これは、例えば、メモリ上にテーブルとして作成される。

【００１１】タイムアウト時間情報設定手段３３は、各スレッドが処理を開始する際に、タイムアウト時間情報管理手段３１にそのスレッドの当該タイムアウト時間管理情報を設定する。

【００１２】タイムアウト検出手段３５は、上記タイムアウト時間情報管理手段３１に設定されたタイムアウト時間管理情報を用いて、タイムアウトの発生したスレッドを検出する。

【００１３】上記構成において、前記タイムアウト検出手段３５は、例えば、タイムアウトの発生したスレッドを削除するような構成にして良い。また、さらに、該スレッドを削除した後に、新たなスレッドを生成するような構成にしても良い。

5

【００１４】また、前記タイムアウト検出手段３５は、例えば、タイムアウトの発生したスレッドを強制的に終了させるような構成にしても良い。図２は、本発明の原理を説明する図（その２）である。

【００１５】この発明に係るサーバ内障害検出装置（以下、第２の発明）は、以下の各手段を備える。サービス手段１００は、クライアントからのリクエストを受信し、該リクエストに対する処理を実行する。タイムアウト時間管理手段１０１は、前記サービス手段１００が行う処理のタイムアウト時間を管理する。監視手段１０２は、前記サービス手段１００を監視する。

【００１６】前記サービス手段１００は、クライアントからのリクエストに対する処理を開始する前に、該処理のタイムアウト時間を前記タイムアウト時間管理手段１０１に設定し、該処理の終了時にタイムアウト時間管理手段１０１の自分が設定したタイムアウト時間を削除する。前記監視手段１０２は、定期的にタイムアウト時間管理手段１０１を参照し、タイムアウト時間内に前記処理が終了しているか否かを検出する。

【００１７】該監視手段１０２は、前記検出処理を開始する前に、例えば、システムカーネルに依頼して自分自身のタイムアウト処理を設定する。そして、該検出処理終了後、該タイムアウト処理を解除する。

【００１８】前記サービス手段１００は、例えば、複数のサービススレッドで構成され、前記監視手段１０２は、例えば、１つの監視スレッドから構成される。また、前記サービス手段１００および前記監視手段１０２をプロセスによって構成するようにしてもよい。

【００１９】前記タイムアウト時間管理手段１０１は、例えば、スレッドＩＤを格納する「スレッドＩＤ」フィールドおよびタイムアウト時間を設定する「タイムアウト時間」フィールドを有する表から構成される。

【００２０】本発明に係るサーバ内障害検出方法は、まず、クライアントからのリクエストを受信する。次に、受信したリクエストに対する処理を実行するサービススレッドのスレッド番号とタイムアウト時間をタイムアウト表に設定する。そして、前記リクエストに対する処理を実行する。該処理終了後、前記タイムアウト表に設定されたスレッド番号とタイムアウト時間を削除する。また、前記タイムアウト表を定期的に参照し、タイムアウトしたサービススレッドが存在するか否かを判別することによってサービススレッドの監視処理を実行する。

【００２１】また、本発明に係るサーバ内障害検出方法は、前記監視処理を開始する前に、監視処理のタイムアウト処理を設定し、前記監視処理の終了後、該タイムアウト処理を解除するようにする。

【００２２】更に、本発明に係るサーバ内障害検出方法は、前記監視処理を終了後、次回の監視処理を始める前に、該監視処理を所定の時間スリープさせる。更に、本発明に係るサーバ内障害検出方法は、サーバが起動時

6

に、初期化スレッドを生成し、該初期化スレッドが、前記タイムアウト表を初期化し、前記監視処理を行う監視スレッドを生成し、クライアントからリクエストを受け付けるリクエスト受付ポートを作成し、該リクエストに対する処理を行うサービススレッド群を生成する。

【００２３】

【作用】第１の発明では、タイムアウト時間情報設定手段３３が、各スレッドが処理を開始する際に、そのスレッドの当該タイムアウト時間情報管理情報を、タイムアウト時間情報管理手段３１に設定する。

【００２４】タイムアウト検出手段３５は、上記タイムアウト時間情報管理手段３１に設定されたタイムアウト時間管理情報を用いて、タイムアウトの発生したスレッドを検出する。

【００２５】したがって、スレッドが処理を開始する毎に、カーネルに対してタイマ処理用のシステムコールを発行せずに、該スレッドのタイムアウトの検出することが可能になり、システムコール実行のためのオーバヘッドが減少する。このため、クライアントに対するサーバのレスポンスが向上する。

【００２６】第２の発明では、前記サービス手段１００は、リクエストに対する処理を開始する前に、該処理のタイムアウト時間をタイムアウト時間管理手段１０１に設定し、前記処理を終了すると、タイムアウト時間管理手段１０１の自分が設定したタイムアウト時間を削除する。

【００２７】これにより、前記監視手段１０２は、定期的にタイムアウト時間管理手段１０１を参照し、例えば、設定されているタイムアウト時間と、現在の時刻を比較することにより、前記処理が設定されたタイムアウト時間内に終了しているか否かを検出することができる。　したがって、サービス手段１００が、リクエストに対する処理を開始する毎に、タイムアウト処理の設定をシステムカーネルに対するシステムコールによって行う必要がなくなり、システムコールの実行回数を減らせる。

【００２８】

【実施例】以下、本発明の実施例について、図面を参照しながら詳細に説明する。図３は、本発明の一実施例の概略を説明する図である。同図に示すように、本実施例のサーバ（サーバプロセス）２０は、クライアント２１ａまたは２１ｂ（以下クライアント２１）からのリクエスト２８を受け付けるポート２２と、クライアントからのリクエスト２８を処理するサービススレッド群２３と、サービススレッド群２３の動作を監視する監視スレッド２４と、各サービススレッド２３ａ～２３ｃによってタイムアウト時間が記入されるタイムアウト表２５と、サーバ２０の初期化を行う初期化スレッド２６から構成される。

【００２９】本実施例では、複数のサービススレッド２

3a～23cを用意することにより、複数のクライアントからのリクエストを並行して処理することができる。なお、本実施例では、サービススレッド群23は、3つのサービススレッド23a～23cから構成されるが、サービススレッドの数は、サーバ20が受け付けるリクエストの個数に応じて動的に増減させることも可能であり、また、システム構成等に応じて変えるようにしてもよい。

【0030】本実施例においては、オペレーティングシステム（OS）に、マイクロカーネル27でOSの基本的な機能を実現し、このマイクロカーネル27上にマイクロカーネル27が提供するプリミティブなインタフェースを使用して、例えば、UNIXをはじめとする標準的なOSをサブシステムサーバとして複数実装するマイクロカーネル方式を採用している。前記サーバ20は、このサブシステムサーバの1つの例であり、マイクロカーネル27上でサーバとして動作するプロセスである。

【0031】ここで、スレッドとプロセスについて、サービススレッド23a～23cとサーバ20を例にして説明する。サービススレッド23a～23cは、プロセッサ割当ての最小単位であり、マイクロカーネル27がスケジューリングの対象とする単位である。これに対して、サーバ20は、サービススレッド23a～23cが参照する仮想記憶空間やファイルなどのシステム資源を割り当てる単位である。1個のサーバ20は1個以上のサービススレッドをもつ。

【0032】ポート22は、クライアント21がサーバスレッド群23間の通信のためのチャネルであり、例えば、マイクロカーネル27の中に存在する、メッセージのキュー30として実現されている。すなわち、例えば、クライアント21から送られてきたリクエスト28でまだサービススレッド23a～23cによって処理されていないポート22宛のリクエスト28は該キュー30に保持される。

【0033】図3において、サーバ20のポート22へ届いたクライアント21からのリクエスト28はサービススレッド群23によって処理される。サービススレッド群23の各サービススレッド23a～23cは、クライアント21からのリクエスト28を受信すると、要求された処理のタイムアウト時間と自分のスレッド番号（スレッドID）をタイムアウト表25に書込む。各サービススレッド23a～23cは、クライアント21からのリクエスト28の処理を終了すると、ポート22を介してクライアント21へリプライ29を送信する。そして、タイムアウト表25内の自身が設定したタイムアウト時間及びスレッドIDを消去する。

【0034】図4は、タイムアウト表25の一構成例を示す図である。同図に示すように、タイムアウト表は、「スレッドID」フィールド25aと、「タイムアウト時間」フィールド25bから構成される。「スレッドID」フィールド25aには、各サービススレッド23a～23cを特定するための識別子であるスレッドIDが格納される。「タイムアウト時間」フィールド25bには、タイムアウト時間として、例えば、タイムアウトする時刻を設定する。これは、サービススレッド23a～23cが、「タイムアウト時間」フィールド25bを設定する際、現在の時刻（設定時の時刻）に所定の時間（タイムアウト時間）を加えたものを設定することで行える。現在の時刻は、例えば、不図示のリアル・タイム・クロック等から成るシステム時計から読み出す。

【0035】また、サービススレッド23a～23cは、クライアント21からのリクエスト28の処理を終了すると、「スレッドID」フィールド25aに自身が設定したスレッドIDと、「タイムアウト時間」フィールド25bに自身が設定したタイムアウト時間の2つの情報を消去する。

【0036】図3において、監視スレッド24は、定期的にタイムアウト表25からサービススレッド群23の各サービススレッド23a～23cが設定したタイムアウト時間を読み出して、現在の時刻と比較することにより、設定されたタイムアウト時間までに処理を終了していないスレッドを検出する。監視スレッド24がタイムアウトしたサービススレッドを発見した場合には、該サービススレッドを強制的に終了または消滅させて、クライアント21にエラーを通知する。

【0037】監視スレッド24は、タイムアウトしたサービススレッドの検出処理を行う前に、マイクロカーネル27がシステムコールとして提供するタイマルーチンを使って、自身のタイムアウト時間を設定する。これによって、監視スレッド24自身が無限ウエイトや無限ループに陥ることを避ける。

【0038】次に、上記各スレッドの処理手順を図5～図7に示すフローチャートを参照しながら説明する。なお、以下に示すフローチャートの説明中、S1、S2、・・・・は処理手順（ステップ）の番号を示す。

【0039】図5は実施例における初期化スレッド26の処理手順を説明するフローチャートである。マイクロカーネル27によって、サーバ20が起動されると、初期化スレッド26が生成される。サーバ20によって生成された初期化スレッド6は、まず、タイムアウト表25を初期化する（S1）。次に、監視スレッド24を生成する（S2）。そして、ポート22を作成する（S2）。続いて、サービススレッド群23を生成する（S4）。初期化スレッド26は、初期化処理が終了すると、自分自身を消滅させる（S5）。なお、本実施例では、初期化処理終了後、初期化スレッド26を消滅させるようにしているが、初期化スレッド26が、初期化処理終了後、監視スレッド24として動作するようにしてもよい。

【0040】図6は実施例におけるサービススレッド群

9

２３の各サービススレッド２３ａ～２３ｃの処理手順を説明するフローチャートである。サービススレッド群２３の各サービススレッド２３ａ～２３ｃは初期化スレッド２６によって生成された後、クライアント２１からのリクエスト２８を処理する。

【００４１】サービススレッド群２３の各サービススレッド２３ａ～２３ｃはポート２２を介してクライアント２１からのリクエスト２８を受信すると（Ｓ１０）、リクエストされた処理のタイムアウト時間と自身のスレッドＩＤをタイムアウト表２５に記入する（Ｓ１１）。そして、リクエスト２８の処理を開始する（Ｓ１２）。該リクエスト２８の処理が終了すると、クライアント２１にその処理結果を返す（Ｓ１３）。そして、タイムアウト表２５から自身が記入したタイムアウト時間とスレッドＩＤを削除して（Ｓ１４）、クライアントからの次のリクエスト２８を待つ。

【００４２】図７は実施例における監視スレッド２４の処理手順を説明するフローチャートである。監視スレッド２４は、初期化スレッド２６によって生成された後、図７に示す手順でサービススレッド群２３の動作を監視する。

【００４３】監視スレッド２４は、まず、マイクロカーネル２７がシステムコールとして提供するタイマルーチンを用いて自身のタイムアウト時間を設定し、マイクロカーネル２７に対してタイムアウト処理を依頼する（Ｓ２０）。次に、タイムアウト表２５からサービススレッド群２３の各サービススレッド２３ａ～２３ｃが記入したタイムアウト時間を読み出す（Ｓ２１）。該読み出したタイムアウト時間と現在の時刻を比較することによって、タイムアウトしたサービススレッドが存在するか否かを判別する（Ｓ２２）。該判別によって、タイムアウトしたサービススレッドを検出した場合には（Ｓ２２、ＹＥＳ）、該タイムアウトしたサービススレッドを削除する（Ｓ２３）。そして、該削除されたサービススレッドが受信したリクエスト２８の送信元クライアント２１にエラーをポート２２を介して通知する（Ｓ２４）。監視スレッド２４は、削除した数のサービススレッドを新規に生成して、サービススレッドを補充する（Ｓ２５）。

【００４４】該ステップＳ２５の処理後、若しくは、上記ステップＳ２２にてタイムアウトしたサービススレッドを検出しなかった場合、監視スレッド２４は一定時間スリープする（Ｓ２６）。監視スレッド２４のプライオリティは一般に高く設定されるため、スリープ状態にならないと、サービススレッド２３ａ～２３ｃが動作できなくなったり、監視処理によって必要以上に処理時間が消費されてしまう。また、このスリープ時間を調整することによって、監視スレッド２４の定周期起動の周期、すなわち、サービススレッド２３ａ～２３ｃのタイムアウト監視処理の周期を変えられる。

10

【００４５】上記スリープ時間が経過後、自身が設定したタイムアウト処理の解除をマイクロカーネル２７へ依頼する（Ｓ２７）。その後、監視スレッド２４は再び、上記処理Ｓ２０～Ｓ２７を繰り返す。

【００４６】図８は、本実施例のサーバ２０の動作を、クライアント２１とマイクロカーネル２７の動作とあわせて説明する図である。まず、マイクロカーネル２７によってサーバ２０が起動されると（Ｓ１００）、サーバ２０は、初期化スレッド２６を生成する。初期化スレッド２６は、生成されると、タイムアウト表２５を初期化する（Ｓ１０１）。次に、監視スレッド２４の生成をマイクロカーネル２７へ依頼する（Ｓ１０２）。該依頼を受けたマイクロカーネル２７は、監視スレッド２４を生成する（Ｓ１０３）。

【００４７】続いて、初期化スレッド２６は、リクエスト受付ポート２２の作成をマイクロカーネル２７へ依頼する（Ｓ１０４）。該依頼を受けたマイクロカーネル２７は、リクエスト受付ポート２２を作成する（Ｓ１０５）。そして、初期化スレッド２６は、サービススレッド群２３の生成をマイクロカーネル２７へ依頼する（Ｓ１０６）。該依頼を受けたマイクロカーネル２７は、サービススレッド２３ａ～２３ｃから成るサービススレッド群２３を生成する（Ｓ１０７）。初期化処理が終了すると、初期化スレッド２６は、自分自身、すなわち、初期化スレッド２６の消滅をマイクロカーネル２７へ依頼する（Ｓ１０８）。該依頼を受けたマイクロカーネル２７は、初期化スレッド２６を消滅させる（Ｓ１０９）。

【００４８】クライアント２１は、サーバ２０に対してサービス処理を依頼する必要が生じると、該サービス処理を依頼するリクエストメッセージを、サーバ２０のポート番号を指定して送信する（Ｓ１１０）。該メッセージを受信すると、マイクロカーネル２７は、該メッセージをサーバ２０のポート２２が有する前記受信キュー３０につなぐ（Ｓ１１１）。

【００４９】サービススレッド群２３の各サービススレッド２３ａ～２３ｃは、前記ステップＳ１０７にて生成されると、まず、サーバ２０のポート２２が有する前記受信キュー３０からリクエストメッセージを読み出す（Ｓ１１２）。そして、自分のスレッドＩＤ及びサービス処理のタイムアウト時間をタイムアウト表２５に記入する（Ｓ１１３）。タイムアウト表２５への記入後、リクエストされたサービス処理を開始する（Ｓ１１４）。

【００５０】該サービス処理が終了すると、サービススレッド２３ａ～２３ｃは、クライアント２１へのリプライ２９の送信をマイクロカーネル２７へ依頼する。該依頼を受けたマイクロカーネル２７は、リプライメッセージをクライアント２１宛の前記送信キューにつなぐ処理を行う（Ｓ１１６）。クライアント２１は、上記送信キューに格納されているリプライメッセージを受信する（Ｓ１１７）。

11

【００５１】クライアント２１へのリプライ処理が終了すると、サービススレッド２３ａ～２３ｃは、タイムアウト表２５から自分が記入したエントリを削除する（Ｓ１１８）。そして、上記ステップＳ１１２～Ｓ１１８を繰り返す。

【００５２】一方、監視スレッド２４は、前記ステップＳ１０３にて生成されると、まず、自分自身のタイムアウト時間を設定し、タイムアウト処理をマイクロカーネル２７へ依頼する（Ｓ１１９）。該依頼を受けたマイクロカーネル２７は、監視スレッドに対してタイムアウト時間を設定する（Ｓ１２０）。そして、監視スレッド２４は、タイムアウト表２５を参照しタイムアウトしたサービススレッドがあるか否かを判別することによって、サービススレッド２３ａ～２３ｃの監視処理を行う（Ｓ１２１）。監視処理終了後、監視スレッド２４は、所定時間スリープを行う（Ｓ１２２）。そして、前記ステップＳ１１９で依頼したタイムアウト処理の解除をマイクロカーネル２７へ依頼する（Ｓ１２３）。該依頼を受けたマイクロカーネル２７は、タイムアウト処理の解除処理を行う（Ｓ１２４）。

【００５３】なお、上記実施例では、初期化スレッド２６によって、予めサービススレッド２３ａ～２３ｃを生成しているが、リクエストが届いた段階で、メッセージハンドラー等によって必要なサービススレッドを動的に生成するようにしてもよい。

【００５４】また、上記実施例においては、クライアントからの依頼をスレッドによって処理するようにしているが、スレッドの代わりにプロセスによって処理するようにすることも可能である。

【００５５】また、サービススレッド２３ａ～２３ｃが、「タイムアウト時間」フィールド２５ｂに所定の時間（タイムアウト時間）を設定し、監視スレッド２４が、定期的に該時間から適当な時間を減算（デクリメント）し、結果が零より大きいか否かによって、設定されたタイムアウト時間までに処理を終了していないサービススレッドを検出するようにしてもよい。

12

【００５６】本発明は、上記実施例以外に、マルチプロセッサ環境での並列処理システムやネットワークで接続された複数の計算機から成る分散処理システム等に実装することも可能である。また、本発明は、マイクロカーネル方式のＯＳのみに限定されるものでもなく、それ以外のＯＳを用いたクライアント／サーバモデルのデータベースシステムなどの各種アプリケーションシステムなどにも適用可能なものである。

【００５７】

【発明の効果】以上、詳細に説明したように、本発明によれば、複数のクライアントからのリクエストを並列に処理するために、サーバ内のサービススレッド数を増やしても、タイムアウト設定のためのシステムコールの実行回数は増加しないので、サーバのレスポンス性能の低下を防ぐことができる。

【図面の簡単な説明】

【図１】本発明の原理図（その１）である。

【図２】本発明の原理図（その２）である。

【図３】本発明の一実施例の概要を説明する図である。

【図４】タイムアウト表の一構成例を示す図である。

【図５】初期化スレッドの処理手順を説明するフローチャートである。

【図６】サービススレッドの処理手順を説明するフローチャートである。

【図７】監視スレッドの処理手順を説明するフローチャートである。

【図８】サーバ２０の動作を、クライアント２１とマイクロカーネル２７の動作とあわせて説明する図である。

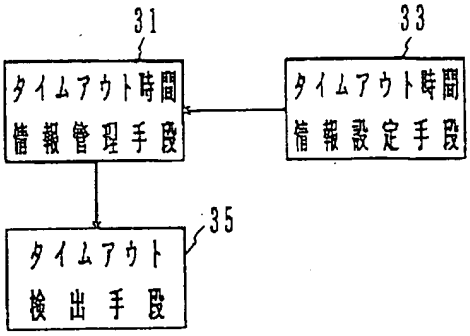
【図９】従来の技術を説明する図である。

【符号の説明】

３１	タイムアウト時間情報管理手段
３３	タイムアウト時間情報設定手段
３５	タイムアウト検出手段
１００	サービス手段
１０１	タイムアウト時間管理手段
１０２	監視手段

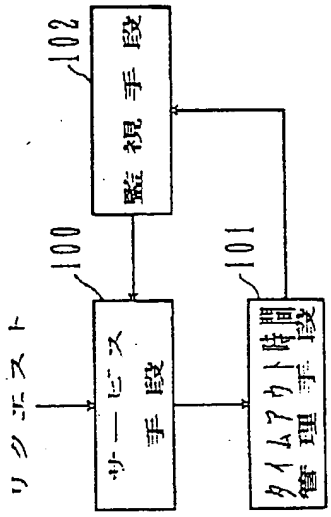
【図1】

本発明の原理図（その1）



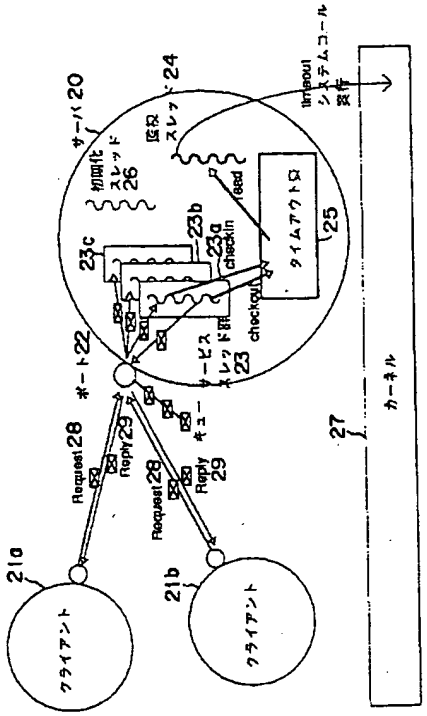
【図2】

本発明の原理図（その2）



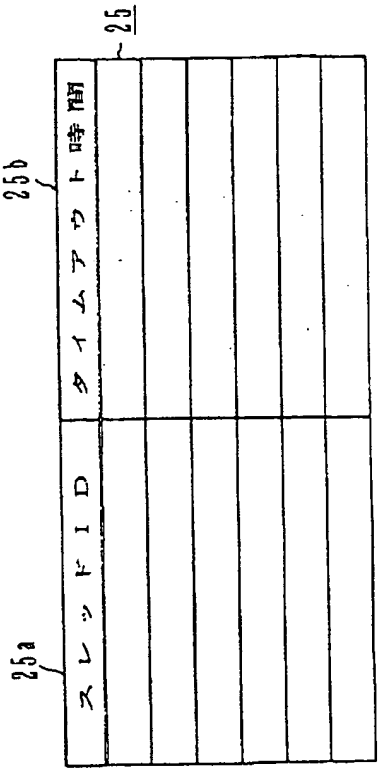
【図3】

本発明の一実施例の概要を説明する図



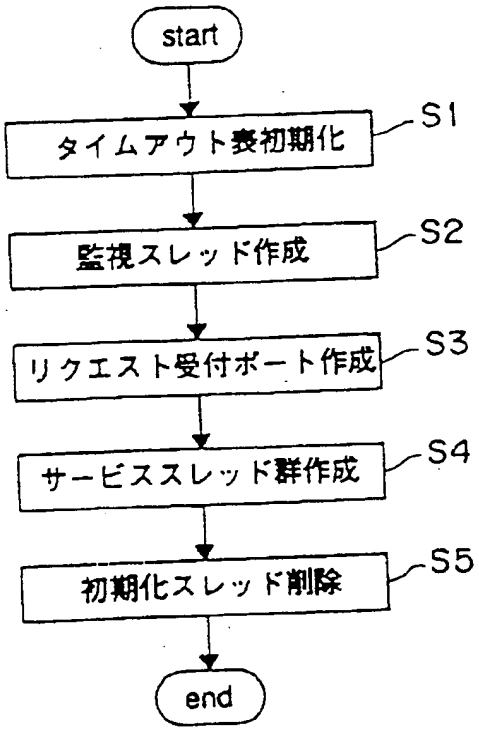
【図4】

タイムアウト表の一構成例を示す図



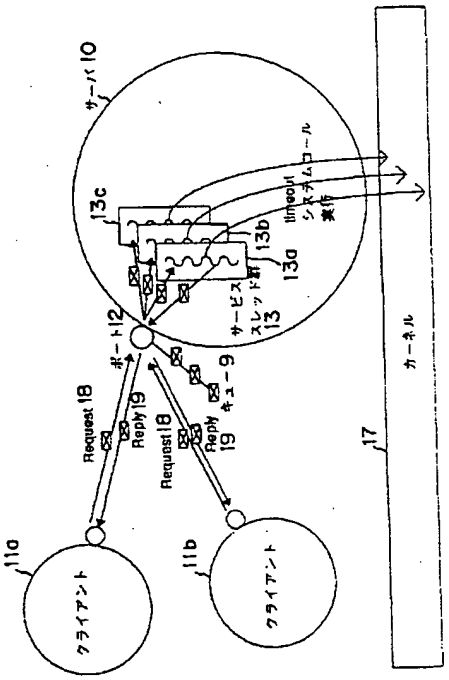
【図5】

初期化スレッドの処理手順を説明するフローチャート



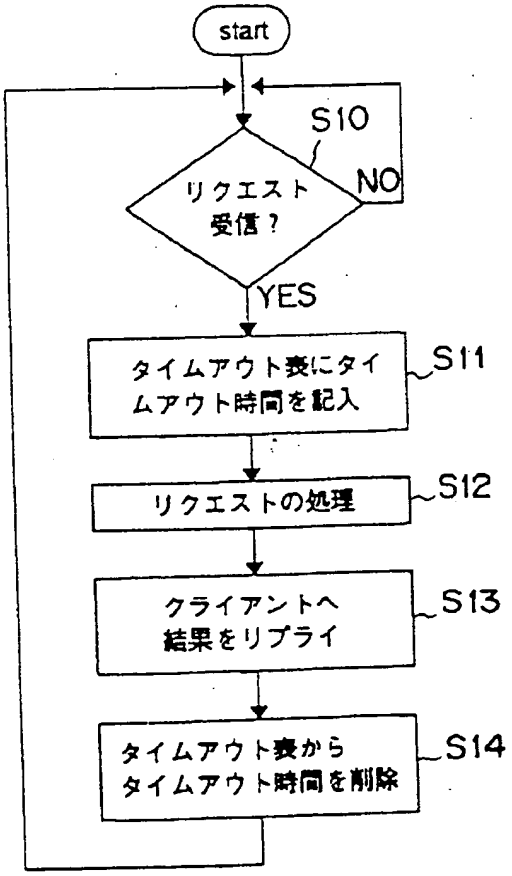
【図9】

従来の技術を説明する図



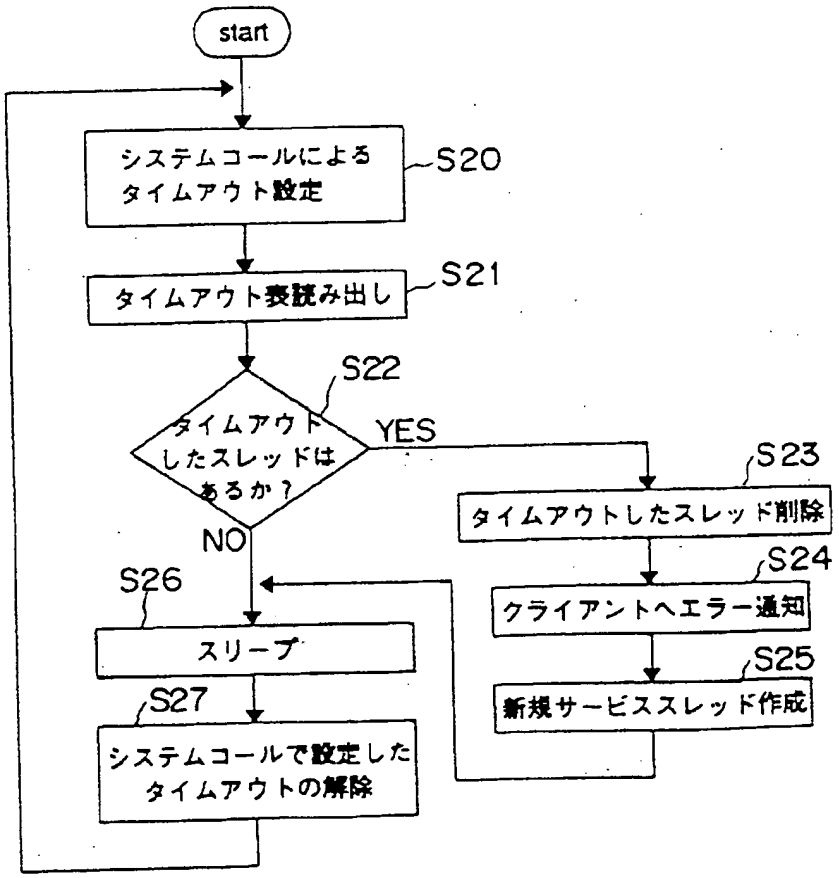
【図6】

サービススレッドの処理手順を
説明するフローチャート



【図7】

監視スレッドの処理手順を説明するフローチャート



【図8】

サーバ20の動作を、クライアント21とカーネル27の動作とあわせて説明する図